Spatial-Temporal Scientific Data Clustering via Deep Convolutional Neural Network

Jianxin Sun[1], Chunxia Wu[2], Yufeng Ge[2], Yusong Li[3], Hongfeng Yu[1]
[1] Department of Computer Science and Engineering
[2] Department of Biological Systems Engineering
[3] Department of Civil and Environmental Engineering
University of Nebraska-Lincoln, Lincoln, NE, USA

***Abstract*—We explore the usage of deep convolutional neural network for clustering the time steps of a spatial-temporal scientific dataset. Our approach first takes the scientific dataset as training data and trains a deep convolutional autoencoder. A low-dimensional feature space or latent space can be extracted by inferencing the encoding part of the network. As a result, each time step is transformed into a feature descriptor that can be compared with each other in the feature space. In this way, we can cluster time steps according to their feature descriptors, and each group of time steps has a similar characterization. We demonstrate the effectiveness of our approach using a real-world simulation dataset of water contamination. Multiple variables and their combinations of this dataset are fed into our approach. The trained network enables the clustering of the time steps and facilitates scientists to examine their large spatial-temporal datasets.**

***Index Terms*—spatial-temporal scientific data, clustering, deep convolutional neural network, autoencoder, feature descriptor**

I. Introduction

The advances in observation and computing techniques enable scientists to obtain datasets with higher spatial-temporal resolutions, leading to detailed discoveries in various domains. However, these datasets are typically multivariate and time-varying. It becomes an increasingly challenging problem for scientists to effectively and efficiently extract the essential information from these vast amounts of datasets.

A feasible way is to cluster the time steps of a dataset into different groups, and each group contains a set of continuous time steps that have a similar characterization of data during the period. This way can allow scientists to explore data in a hierarchical fashion such that they can quickly gain an overview of a dataset, and effectively select different groups of time steps for a closer examination.

Researchers have developed several approaches to achieve this goal. Some methods are based on the comparison between time steps [1], [2], which mostly rely on the local information, and may not capture dynamics across a wider period. Some methods are based on the information theory [3], [4], which is more sophisticated but may incur a high computational cost.

In this paper, we develop an autoencoder based approach to learn feature descriptors of time steps of a spatial-temporal scientific dataset in an unsupervised fashion. Given a dataset, to reduce the computational cost, we extract 2D slices from each volume. In the first training stage, these 2D slices are fed into an autoencoder for training its deep convolutional neural networks. After training, a low-dimensional feature space (or latent space) is generated by the autoencoder. Then, in the inferencing stage, the trained network takes each time step as input and generates the corresponding vectors in the latent space. The generated vectors become a feature descriptor of a time step. We compute the distance of these feature descriptors in the feature space, cluster them into different groups, and thereby facilitate scientists to explore different clusters and examine the details in a cluster. Our preliminary study shows promising results of our approach with a real-world simulation of contamination of groundwater.

II. Related Work

One of the main characters of complex scientific data set is its high dimensionality. Unsupervised learning methods are widely used to perform dimensionality reduction for better clustering the data in a lower-dimensional space. With the emerging of deep learning, many deep learning based methods, such as autoencoder and RBM (Restricted Boltzmann Machine), are proposed and implemented with promising results in various domains [5]–[7]. Autoencoder consists of fully connected dense layers and struggles to provide the best performance to deal with spatial-temporal data sets where values of neighboring data points are highly correlated.

Inspired by the effectiveness of 2D convolutional layers in deep learning network for handling 2D images, 3D convolutional autoencoder becomes a natural unsupervised learning method to deal with 3D data [8]. However, 3D convolution is computational expensive as it has a complexity of $O(N^3)$ where N is the size of the input 3D cube. Some works make a compromise to keep using 2D convolutional autoencoder by extracting a 2D representation from 3D data. For example, Chen et al. [9] directly used single 2D CT scan without considering the spatial relation across neighboring scan in sequence. Zhu et al. [10] simply projected original 3D data into 2D space with inevitable information loss. It is critical to design a effective and efficient learning architecture when dealing with large spatial-temporal scientific dataset.

Recently, deep learning has been exploited in scientific data analysis and visualization. Han et al. [11] used autoencoder to cluster and select streamlines and stream surfaces. Berger et al. [12] employed Generative Adversarial Network (GAN) to facilitate transfer function designs for volume rendering.

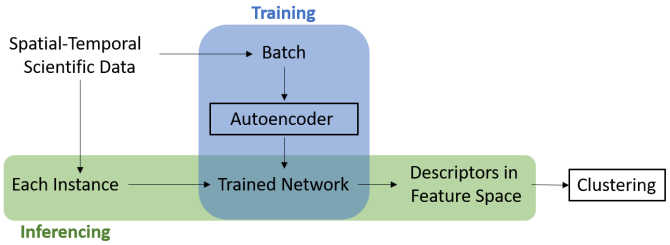


Fig. 1. The major components of our approach.

He et al. [13] presented a deep learning based surrogate model to support the parameter space exploration for ensemble simulations that are visualized in situ.

III. Deep Learning Architecture

In this work, we generate a feature descriptor for each time step of a spatial-temporal scientific data, and measure the similarities among the time steps based on their feature descriptors. Then, we partition the time steps of the dataset into different groups, where each group of time steps has similar feature descriptors.

Figure 1 shows the architecture of our approach that consists of the following major components. First, in the training stage, we train an autoencoder of deep neural network with convolutional and transposed convolutional layers using batches of an input 3D scientific dataset. Through the training, parameters of the network are adjusted by backpropagation such that the network can regenerate the input on the output side. Second, in the inferencing stage, each time step is fed into the encoder of the trained network and is then calculated into a set of low-dimensional vectors (i.e., *feature descriptor*) where the dynamics across the time steps of the dataset can be summarized. Third, we cluster the time steps according to the distributions of their feature descriptors.

A. Data Preprocessing

We consider a 3D time-varying scientific dataset in this work. Each time step D_t of the dataset contains multiple 3D volumes, and each volume $V_t(\alpha)$ corresponds to a variable α. To reduce the computational cost of training, we extract a set of even-spacing 2D slices from each 3D volume at a time step. In this work, there are 8 slices $S_t = \{s_{t_1}, ..., s_{t_8}\}$ for each 3D volume, and we use them as an *instance* to represent the volume. The 2D slices of all time steps are used as the training set. Before being fed into the deep learning autoencoder, all 2D slices are standardized. We scale each slice s_{t_i} such that its size is a square shape, which can simplify the 2D convolutional operation in the deep neural network. For a multivariate dataset, each entry of a 2D slice may contain one or more input values that are specified by users.

B. Feature Space Learning

We exploit autoencoder as the main unsupervised learning method to derive the feature space. An autoencoder is used to extract a low-dimensional feature space [14]. A convolutional autoencoder considers the spatial information of input objects, and it is suitable for effectively extracting a feature space for objects with spatial correlations that are commonly inherent in spatial-temporal scientific datasets. An typical autoencoder consists of *encoder* and *decoder*. A loss function is used to minimize the difference between the input of the encoder and the output of the decoder. Since the value of each position in 3D volume is highly correlated with its neighboring points, we utilize convolutional neural network to utilize the local spatial information.

Figure 2 shows the detailed structure of the autoencoder with 2D convolutional neural network. The input is 2D data array from the slices of the 3D volume in the x-y plane. The 2D convolutional structure is composed of 3 convolutional layers and 3 transposed convolutional layers. We design a feature map with different size for each downstream convolutional layer. The same structure is also used for the 3 transposed convolutional layers. The encoding convolutional network is followed by fully connected layers to extract the feature space or latent space. In our work, we set the latent space as a 2D space. The dimension of the input layer depends on the size of our data. We use *tanh* as the activation function, and the mean squared errors (MSE) as a metric of the loss function.

C. Feature Descriptor Inferencing

After we obtain the trained network, we use it to generate the feature descriptor for each time step through inferencing. For each slice s_{t_i} extracted from a 3D volume, we feed it into the trained encoder and generate a vector $\mathbf{p}_{t_i}$ in the 2D latent space. Therefore, we can obtain 8 2D vectors $\mathbf{p}_{t_1}, ..., \mathbf{p}_{t_8}$ to represent a 3D volume in the latent space.

D. Clustering

These 8 vectors are the feature descriptor of the 3D volume in the latent space. Therefore, we compute the difference $Diff_{mn}$ between two volumes at the time steps m and n as the sum of the distances:

$$Diff_{mn} = \sum_{i=1}^{8} (|\mathbf{p}_{m_i} - \mathbf{p}_{n_i}|), \quad (1)$$

where $|\mathbf{p}_{m_i} - \mathbf{p}_{n_i}|$ is the Euclidean distance between the 2D vectors $\mathbf{p}_{m_i}$ and $\mathbf{p}_{n_i}$. Based on the distance $Diff_{mn}$ between any two time steps m and n, we use the k-means algorithm to cluster all the time steps of a dataset.

IV. Results and Discussion

A. Datasets and Training

We have experimented our approach using a simulation dataset of contamination of groundwater from nitrogen fertilizers [15]. We used three variables, water content, water pressure, and water velocity magnitude, in our experiment. We downsampled the dataset with a dimension of $48 \times 40 \times 8$ (length$\times$width$\times$depth). The dataset contains 143 time steps.

We extracted 8 slices in the x-y plane for each volume. As an input slice is 48×40, we scale it on the y axis from

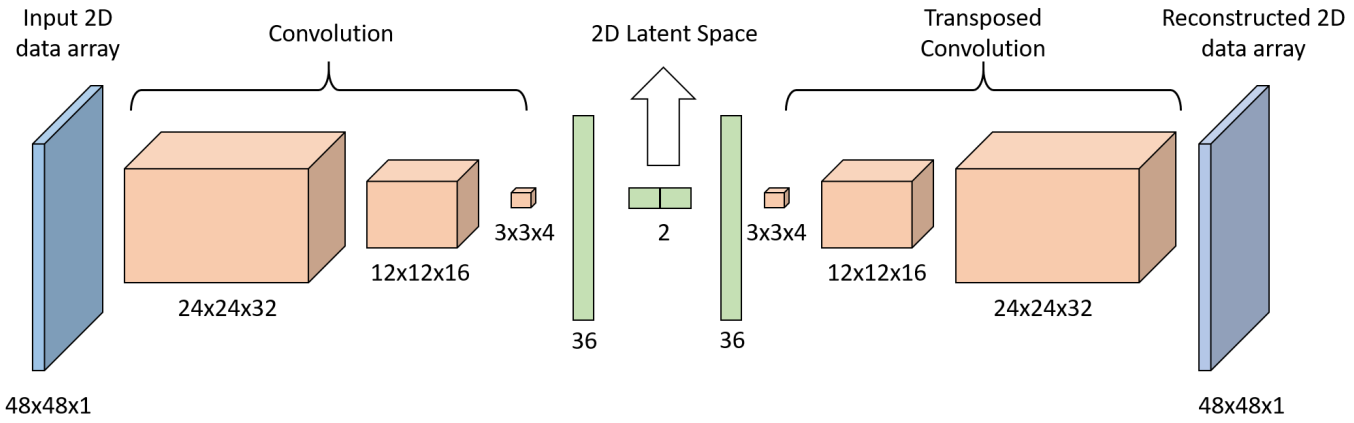
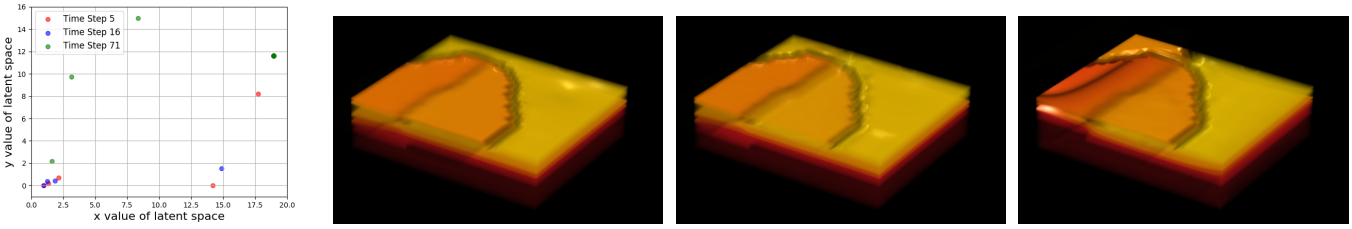


Fig. 2. The network structure of convolutional autoencoder.



(a) (b) (c) (d)

Fig. 3. Example of feature descriptors and volumes: (a) the latent space distribution of feature descriptors of water content in the time steps 5, 16, and 71; (b), (c), and (d) the volume rendering results of the time steps 5, 16, and 71, respectively.

40 to 48 by linear interpolation. The raw dataset has 143 3D volumes. As there are 8 slices in each volume, there are in total 1,144 2D slices as the training dataset. The dimensions of 3 convolutional layers are $24 \times 24 \times 32$, $12 \times 12 \times 16$, and $3 \times 3 \times 4$, respectively. The network is trained for 100 epochs with a learning rate of 0.001.

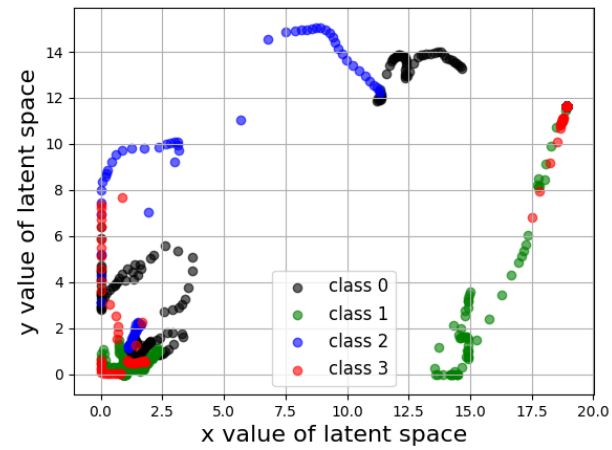
B. Inferencing and Clustering Results

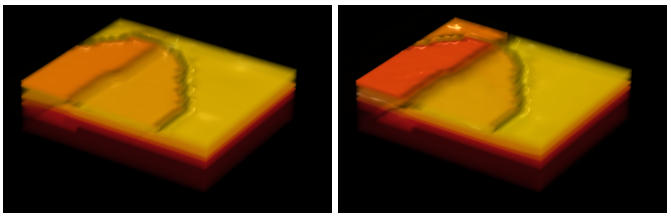
After network training, we fed the 8 slices of each volume into the network, and collect each respective inferencing result together as a feature descriptor of the volume (i.e., 8 2D vectors in the latent space). Figure 3 shows an example of feature descriptors and volumes. The feature descriptors of water content in the time steps 5, 16, and 71 are shown in Figure 3(a), where water content of each time step is represented as a set of 2D vectors. We can see that the vectors of the earlier time steps 5 and 16 are relatively close to each other, while the vectors of the later time step 71 have a distinct distribution. Figure 3(b), (c), and (d) show the volume rendering results of these three time steps, which qualitatively reveal that the water content of the time step 71 has a significant change compared to those two earlier stages.

The top image in Figure 4 shows the latent space distribution of water content's feature descriptors across all time steps. We applied the k-means algorithm to cluster these descriptors into 4 classes using Equation 1 for distance measure. Different classes are labeled with different colors in Figure 4. Correspondingly, all time steps are partitioned into 4 periods, 0-49, 50-65, 66-89, and 90-142, respectively. The volumes within each period have similar feature descriptors. Figure 4 also shows the volume rendering result of the representative time step of each period. We can see the distinct visualization results among these periods.

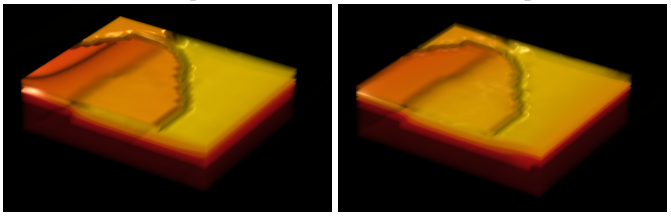
The top image in Figure 5 shows the latent space distribution of water pressure's feature descriptors across all time steps, which is significantly different from water content. We also applied the k-means algorithm to generate 4 clusters. The clustered periods are different from water content as well, and consist of the ranges of 0-50, 51-66, 67-128, and 129-142. The volume rendering results in Figure 5 show the representative time steps of individual periods of water pressure.

Similarly, we also clustered the time steps according to water velocity magnitude. Figure 6 shows the feature descriptors of all time steps with respect to water velocity magnitude in the latent space. Based on their distributions, we also clustered the feature descriptors into 4 groups, and similar time steps can be grouped together according to their feature descriptors, as shown in Figure 6. The clustered periods consist of the ranges of 0-69, 70-80, 81-131, and 132-142. The representative time step of each period is volume rendered at the bottom of



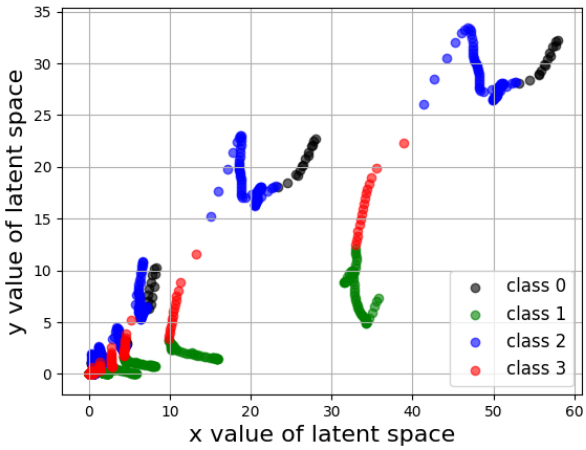


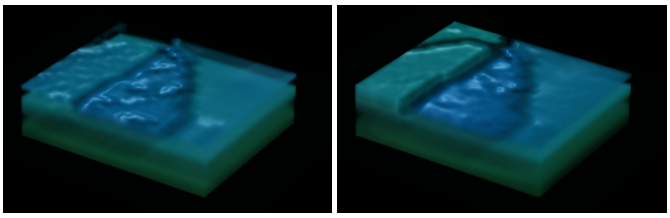
Time Step 12 Time Step 65



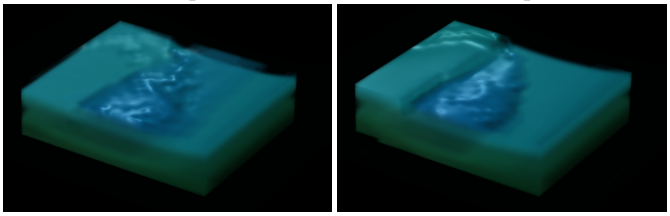
Time Step 71 Time Step 104

Fig. 4. Top: Latent space distribution of water content's feature descriptors across all time steps, which is clustered into 4 classes. Bottom: Volume rendering result of the representative time step of each cluster of water content.





Time Step 26 Time Step 65



Time Step 87 Time Step 132

Fig. 5. Top: Latent space distribution of water pressure's feature descriptors across all time steps, which is clustered into 4 classes. Bottom: Volume rendering result of the representative time step of each cluster of water pressure.

Figure 6.

As described in Section III, our approach can also learn a feature space by incorporating multiple variables in the 2D slices. In this case, each entry of a 2D slice contains multiple values, which can be directly fed into our autoencorder framework. The top image in Figure 7 shows the distribution of the feature descriptors by leaning all three variables, water content, water pressure, and water velocity magnitude, across all time steps. We also clustered them into 4 classes. By considering more variables, the time steps are partitioned into 6 periods associated with 4 clusters, as shown in Table I. We can see that Class 3 has periodically appeared among the time steps. The bottom of Figure 7 shows the volume rendering results of the representative time step of each period. In each plot, water content and pressure are visualized together using multivariate volume rendering technique. We can see the considerable similarity among the time steps of Class 3, although they are distributed among the time steps of other classes.

TABLE I
CLUSTERING OF TIME STEPS USING ALL THREE VARIABLES.

Class	3	0	3	2	3	1
Time Step Range	0-12	13-55	56-65	66-80	81-129	130-142

V. CONCLUSION AND FUTURE WORK

In this position paper, we demonstrate the feasibility to use autoencoder with deep convolutional neural network to cluster the time steps of a scientific dataset. Our preliminary results show that feature descriptors can be learned for individual variable and their combinations for a real-world simulation data. Each time step can be represented as a set of vectors in the feature space. Using this representation, we can quantify the distance between the time steps and cluster them into different groups, where each group has similar patterns. Our visualization results qualitatively reveal the difference and the similarity among different classes. Our approach makes it easy

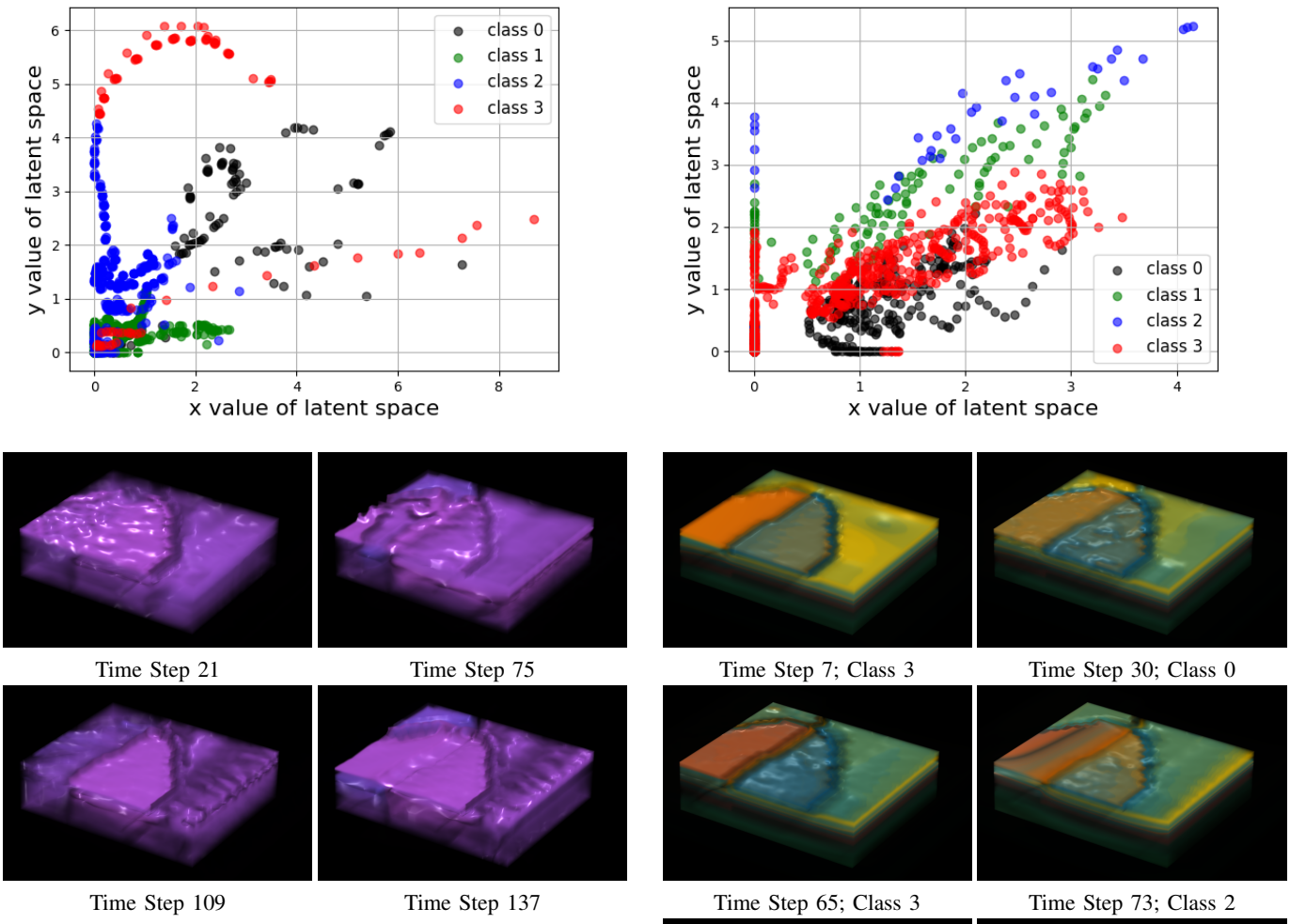


Fig. 6. Top: Latent space distribution of water velocity magnitude's feature descriptors across all time steps, which is clustered into 4 classes. Bottom: Volume rendering result of the representative time step of each cluster of water velocity magnitude.

Fig. 7. Top: Latent space distribution of feature descriptors by learning all three variables, water content, water pressure, and water velocity magnitude, across all time steps, which is clustered into 4 classes. Bottom: Volume rendering result of water content and pressure at the representative time step of each period.

for domain scientists to explore and select time steps from their large-scale spatial-temporal scientific datasets, as well as help researchers design new algorithms for data compression and reduction.

Our preliminary study can be strengthened from multiple aspects. First, we extract 2D slices from 3D volumes as the input for network training in this work. The parameters, such as the number and the orientation of the slices, will be examined in the next stage. We would also like to investigate other techniques to represent 3D volumes (e.g., sampling, multi-resolution representations, etc.), and study their impacts on performance and quality of learning. In addition, we plan to study quantitative measures for the quality of our results, and compare them with other existing approaches. We note some less optimal clustering results in our current work. We will explore the usage of different clustering algorithms (e.g., DBSCAN [16]) with different cluster numbers on feature descriptors. The dimension of the latent space will also be investigated. We will experiment our approach with datasets generated from different domains, and explore possible scientific meanings and discoveries from clustering results with domain experts.

VI. Acknowledgment

This research has been partially supported by the National Science Foundation through grants 1541043 and 14234877,

and by the United States Department of Agriculture through grants 2017-67021-26248 and 2014-67003-22072.

REFERENCES

[1] H.-W. Shen, L.-J. Chiang, and K.-L. Ma, "A fast volume rendering algorithm for time-varying fields using a time-space partitioning (TSP) tree," in *Proceedings Visualization'99 (Cat. No. 99CB37067)*. IEEE, 1999, pp. 371–545.

[2] K.-C. Wang, K. Lu, T.-H. Wei, N. Shareef, and H.-W. Shen, "Statistical visualization and analysis of large data using a value-based spatial distribution," in *2017 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 2017, pp. 161–170.

[3] C. Wang, H. Yu, and K.-L. Ma, "Importance-driven time-varying data visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1547–1554, 2008.

[4] C. Wang, H. Yu, R. W. Grout, K.-L. Ma, and J. H. Chen, "Analyzing information transfer in time-varying multivariate data," in *2011 IEEE Pacific Visualization Symposium*. IEEE, 2011, pp. 99–106.

[5] L. Nie, A. Kumar, and S. Zhan, "Periocular recognition using unsupervised convolutional rbm feature learning," in *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014, pp. 399–404.

[6] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[7] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder." in *Interspeech*, 2013, pp. 436–440.

[8] E. Hosseini-Asl, R. Keynton, and A. El-Baz, "Alzheimer's disease diagnostics by adaptation of 3D convolutional network," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 126–130.

[9] M. Chen, X. Shi, Y. Zhang, D. Wu, and M. Guizani, "Deep features learning for medical image analysis with convolutional autoencoder neural network," *IEEE Transactions on Big Data*, 2017.

[10] Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai, "Deep learning representation using autoencoder for 3D shape retrieval," *Neurocomputing*, vol. 204, pp. 41–50, 2016.

[11] J. Han, J. Tao, and C. Wang, "FlowNet: A deep learning framework for clustering and selection of streamlines and stream surfaces," *IEEE transactions on visualization and computer graphics*, 2018.

[12] M. Berger, J. Li, and J. A. Levine, "A generative model for volume rendering," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 4, pp. 1636–1650, 2018.

[13] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. Nashed, and T. Peterka, "InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations," *IEEE transactions on visualization and computer graphics*, 2019.

[14] Y. Bengio *et al.*, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[15] S. Akbariyeh, S. Bartelt-Hunt, D. Snow, X. Li, Z. Tang, and Y. Li, "Three-dimensional modeling of nitrate-n transport in vadose zone: Roles of soil heterogeneity and groundwater flux," *Journal of contaminant hydrology*, vol. 211, pp. 15–25, 2018.

[16] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.